Metody kompilacji

Laboratorium 1

Piotr Błaszyński

5 maja 2018

Zadania:

- pobrać wszystkie pliki z przykładami: `http://detox.wi.ps.pl/pb/tk/wszystkieZ.zip`
- przetestować kompilację i uruchomienie kodu z katalogów z1, z2, z3 (instrukcja na końcu),
- uruchomienie polega na podaniu jako argumentu pliku inX.txt, pliki znajdują się w odpowiednich katalogach,
- dodatkowo przetestować pracę w trybie interaktywnym,
- zmodyfikować pliki wejściowe w z2, tak aby się „kompilowały" obydwa, ew. można zmodyfikować plik z analizatorem (z2.l), jest to zadanie na zapoznanie się ze szkieletem budowy analizatora leksykalnego, należy sprawdzić jakie leksemy obsługuje zaprezentowany analizator leksykalny, i usunąć z pliku źródłowego nieobsługiwane elementy,
- zapoznać się z treścią Makefile w z3,
- pobrać emulator procesora MIPS - MARS `http://courses.missouristate.edu/KenVollmar/mars/download.htm` (alternatywnie QtSpim), uruchamianie emulatora poprzez maszynę wirtualną javy (przykład na końcu),
- uruchomić przykładowy program `http://courses.missouristate.edu/KenVollmar/mars/CCSC-CP%20material/row-major.asm`

Zadanie domowe: Opracować projekt **własnego** języka. Łatwiej zrobić kompilator języka tradycyjnego, unikamy konstrukcji z języków ezoterycznych. Oprócz samych konstrukcji proszę przygotować kilka plików z programami w swoim języku (testujących poniższe konstrukcje). Wymagane konstrukcje (ocena - wymagania):

- 3.0 - typy int i double - stałe (literały) i zmienne tych typów, wyrażenia arytmetyczne (=, +, -, *, /), prosty if (bez else i co najwyżej jedno zagnieżdżenie), wypisywanie int i double, wpisywanie int i double z konsoli
- 3.5 - pętla for lub while, typ string (tylko wypisywanie, brak operacji),
- 4.0 - tablice jednowymiarowe, złożony if (dużo zagnieżdżeń) z else
- 4.5 - tablice wielowymiarowe,
- 5.0 - dynamiczna alokacja tablic jednowymiarowych lub proste funkcje (procedury bez parametrów) (ew. funkcje z parametrami i wartością zwracaną),

Zadanie sprawdzam za tydzień, w wersji elektronicznej.

Wywołanie kompilacji w pierwszych dwóch katalogach:

```
flex zX.l #powstaje plik lex.yy.c
gcc -c lex.yy.c #powstaje plik lex.yy.o
gcc lex.yy.o -o nazwa_mojego_kompilatora -ll # powstaje
   plik nazwa_mojego_kompilatora
```

Uruchamianie:

```
./nazwa_mojego_kompilatora #uruchomienie w trybie
   interaktywnym
./nazwa_mojego_kompilatora < in1.txt #uruchomienie z
   podaniem na stdin pliku in1.txt
./nazwa_mojego_kompilatora in1.txt #uruchomienie z
   podaniem w argv[1] in1.txt (trzeba obsluzyc w main)
```

Tryb interaktywny kończymy zawsze naciskając:

```
Ctrl+D
```

Z pliku Makefile korzystamy wpisując:

```
make
```

Wywołanie bisona (korzystamy z Makefile, ale trzeba wiedzieć co się w nim dzieje):

```
bison -d def.y #powstaje plik def.tab.c i def.tab.h (w
   wersji C++ def.tab.cc i def.tab.hh)
gcc -c def.tab.c #powstaje plik def.tab.o
gcc lex.yy.o def.tab.o -o nazwa_mojego_kompilatora -ll #
    powstaje plik nazwa_mojego_kompilatora
```

Przykładowe uruchamienie emulatora MARS (można również uruchamiać w środowisku graficznym, trzeba wcześniej ustawić uprawnienia pliku do wykonywania - chmod +x Mars4_5.jar:

```
java -jar Mars4\_5.jar
```